AD 699 950

DIVISIBLE AND MOVABLE ACTIVITIES IN CRITICAL PATH ANALYSIS

William S. Jewell

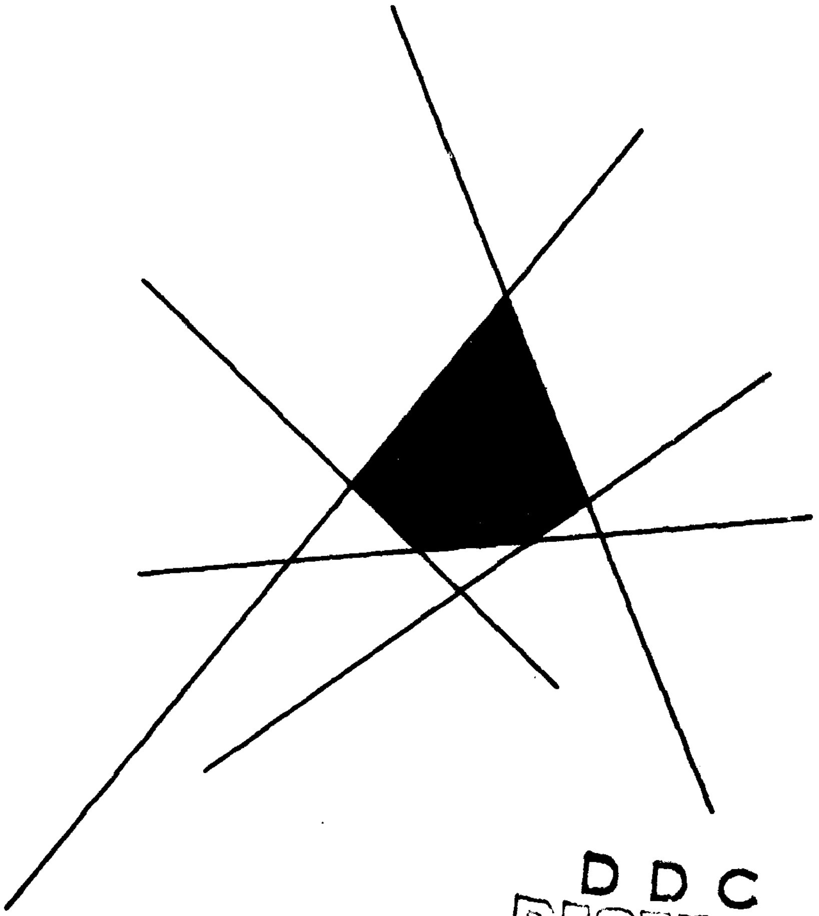
California University
Berkeley, California

November 1969

ORC 69-34
NOVEMBER 1969

DIVISIBLE AND MOVABLE ACTIVITIES IN CRITICAL PATH ANALYSIS

by
WILLIAM S. JEWELL



OPERATIONS
RESEARCH
CENTER

COLLEGE OF ENGINEERING
UNIVERSITY OF CALIFORNIA · BERKELEY

50

DIVISIBLE AND MOVABLE ACTIVITIES IN CRITICAL PATH ANALYSIS

by

William S. Jewell
Department of Industrial Engineering
and Operations Research
University of California, Berkeley

and

TEKNEKRON, Inc.
Berkeley, California

Presented at the Second International Congress on
Project Planning by Network Analysis, Amsterdam, 6-10 October 1969.

NOVEMBER 1969 ORC 69-34

This research has been partially supported by the Office of Naval Research under Contract Nonr-222(83), the National Science Foundation under Grant GP-8695, the National Science Foundation under Grant GK-1684 and the U. S. Army Research Office-Durham under Contract DA-31-124-ARO-D-331 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Travel and presentation support by the Office of Naval Research and TEKNEKRON, Inc. are gratefully acknowledged.

ABSTRACT

Certain jobs in large projects do not have a unique "location" in the critical path network; they may be moved into certain slack intervals, for example, or may even be divisible into smaller subtasks, and "tucked in" at several locations. An example is the painting activity during aircraft maintenance and overhaul which may logically be performed at several points during the overhaul cycle.

In a previous paper [4], an analysis was given of a model in which a single job can be divided up in any manner among an arbitrary number of locations; the resulting algorithm was of the optimal network flow type, which can be simply and efficiently solved using available computer codes. In the first part of the present paper, this model is extended to *multiple* jobs of divisible type. The general approach is via the decomposition method of linear programming; however, the resulting algorithm is again fairly simple. Optimal cost-time solutions, possibly infinite, (or optimal network flow solutions, possibly infeasible) are generated by known algorithms. The resulting schedules or cuts are then combined in a simple, special-structure linear program whose dimensionality is equal to the number of divisible groups. Bounds on the nonintegrality of the final allocations can also be determined.

When these special jobs can only be moved about the network in their entirety, or in certain indivisible modules, the problem takes on the form of an integer program. In the second part of the paper, a branch-and-bound procedure will be given for the problem of movable activities, together with efficient heuristics for arbitrating and bounding these locations, using only the ordinary critical-path algorithm.

Examples are given for both models.

DIVISIBLE AND MOVABLE ACTIVITIES IN CRITICAL PATH ANALYSIS

by

William S. Jewell

0. INTRODUCTION

Certain jobs in large projects do not have a unique "location" in the critical path network; they may be moved into certain slack intervals, for example, or may even be divisible into smaller subtasks, and "tucked in" at several locations. An example is the painting activity during aircraft maintenance and overhaul which may logically be performed at several points during the overhaul cycle.

In a previous paper [4], an analysis was given of a model in which a single job can be divided up in any manner among an arbitrary number of locations; the resulting algorithm was of the optimal network flow type, which can be simply and efficiently solved using available computer codes. In the first part of the present paper, this model is extended to multiple jobs of divisible type giving a decomposition algorithm with special structure in both the master and subprograms.

Another possibility is that such a task cannot be subdivided as finely as desired, but can only be *moved* around in certain modules, or in its entirety. In the second part of this paper, a branch-and-bound algorithm is developed which uses the ordinary critical path algorithm in an efficient manner.

1. FORMULATION

Assume that a *project network* of A arcs (*activities*) and N nodes (*events*) is given. Arc ij represents activity beginning at node i and ending at node j, nodes 1 and N represent *Start* and *Finish* of the project, respectively. *Activity durations* $\{T_{ij}\}$ are given for each arc, and the problem is to select potentials (*event epochs*) $\{v_i\}$ for each node which minimize the *total project duration,* $F = v_N - v_1$. As usual, we assume all T_{ij} finite, that no directed cycle exists with positive sum of T_{ij}, and at least one directed path exists from node 1 to N.

Additionally, we assume there are D *divisible activities* which require duration T_d, $(d = 1,2, \ldots, D)$. These durations may be divided up in arbitrary amounts and carried out at any or all of a certain number of "locations" in the network, specified in advance for each such activity. For simplicity in notation, artificial arcs are added, if necessary, so that an arc is either a location for a single divisible task, or for none. In other words, partition the class of all arcs, $\mathcal{A}$, into $D + 1$ exclusive subsets, $\mathcal{A}_o, \mathcal{A}_1, \ldots, \mathcal{A}_D$, and read "$ij \in \mathcal{A}_d$" as "activity ij belongs to divisible class d"; conversely, "$d \in \mathcal{A}^{-1}(ij)$" reads "$d$ is the (unique) class to which ij belongs." Let A_d be the number of arcs in class d, so that $A_o + A_1 + \ldots + A_D = A$.

Finally, define t_{ij} as the additional duration expended on divisible work at location ij; the *initial primal problem* is then:

(1a) $$\text{Min } F = v_N - v_1$$

(1b) $$v_j - v_i - t_{ij} \geq T_{ij} \qquad \forall\ ij \in \mathcal{A}$$

(1c) $$t_{ij} \geq 0$$

(1d) $$\sum_{ij \in \mathcal{A}_d} t_{ij} = T_d \qquad d = 1,2, \ldots, D .$$

It is immaterial whether we set $T_o = 0$, or drop the t_{ij} from arcs in A_o. The constraint matrix for the example of Section 8 is shown in Figure 1. The upper part of the matrix is the same as that of a cost-time critical path (CPM) problem [3], [4] where the objective is, typically:

$$\text{(1a')} \qquad \text{Min } F = Q(v_N - v_1) - \sum_{ij \in A} C_{ij} t_{ij} + \text{fixed terms} .$$

Here C_{ij} is a unit savings gotten by "lengthening" ij by an amount t_{ij} above the "crash" duration T_{ij}; usually there is an upper bound on t_{ij} to limit paid lengthening to some "normal" duration. Q is a Lagrange multiplier varied during the course of the cost-time algorithm, which has a dual interpretation of exogenous flow.

The constraints (1d), on the other hand, are like the "bundle constraints" of multi-commodity flow problems [3], [6], [9], except that here they are adjoined to the *transpose* of the usual *Kirchoff flow matrix.*

The *dual initial problem* is:

$$\text{(2a)} \qquad \text{Max } L = \sum_{ij \in A} T_{ij} x_{ij} + \sum_{d=1}^{D} T_d y_d$$

$$\text{(2b)} \qquad \sum_j (x_{ji} - x_{ij}) = \begin{cases} -1 & i = 1 \\ +1 & i = N \\ 0 & \text{otherwise} \end{cases}$$

$$\left.\begin{array}{ll} \text{(2c)} & x_{ij} \geq 0 \\ \text{(2d)} & x_{ij} \geq y_{A^{-1}(ij)} \end{array}\right\} \quad ij \in A$$

which is the usual dual *longest-route problem*, except for a possible profit for simutaneously increasing several lower bounds, rather like an optimal capacity contraction problem.

It is the confluence of these different, but special models that makes the analysis of (1) interesting from a pedagogical point of view, providing we can retain the computational simplicity of these special structures.

2. DECOMPOSITION

We shall solve (1) by using the *decomposition method* (see [2] or [10, Chapter 10]), with (1d) as the *master problem* and (1b), (1c) as the *subproblem*. The master will be solved by the simplex method (with some simplifications), and the subproblem will be solved by special CPM or optimal flow algorithms. For convenience, we review the principles of this method in our notation.

The set of all solutions to (1b), (1c) is a convex polytope with both finite and unbounded regions; in fact, the unbounded region usually turns out to be important in the application of the algorithm. Let $\left\{v_i^k;t_{ij}^k\right\}$ $(k = 1,2, \ldots, K)$ be the set of all *extremal solutions* to the finite region (a polyhedron) and $\left\{w_i^\ell;u_{ij}^\ell\right\}$ $(\ell = 1,2, \ldots, L)$ be the *extremal rays* bounding the infinite part of the solution space (a convex polyhedral cone). From a well-known theorem of convex sets [10], *any* feasible set of solutions $\{v_i;t_{ij}\}$ to (1b), (1c) can be represented as follows:

$$\text{(3a)}\quad v_i = \sum_k \lambda_k v_i^k + \sum_\ell \theta_\ell w_i^\ell \ ; \ t_{ij} = \sum_k \lambda_k t_{ij}^k + \sum_\ell \theta_\ell u_{ij}^\ell$$

$$\text{(3b)}\quad \sum_k \lambda_k = 1 \ ; \ \lambda_k \geq 0 \ (k = 1, \ldots, K) \ ; \ \theta_\ell \geq 0 \ (\ell = 1,2, \ldots, L) \ .$$

Define new constants for each extremal point and ray as follows:

$$\text{(4)}\quad F^k = v_N^k - v_1^k \ ; \ G^\ell = w_N^\ell - w_1^\ell \ ; \ T_d^k = \sum_{ij\varepsilon A_d} t_{ij}^k \ ; \ U_d^\ell = \sum_{ij\varepsilon A_d} u_{ij}^\ell \ .$$

We find the new form of the master problem in the *mixing variables* $\{\lambda_k,\theta_\ell\}$:

(5a) $$\text{Min} \quad F = \sum_k F^k \lambda_k + \sum_\ell G^\ell \theta_\ell$$

(5b) $$\sum_k \lambda_k = 1$$

(5c) $$\sum_k T_d^k \lambda_k + \sum_\ell U_d^\ell \theta_\ell = T_d \quad (d = 1,2, \ldots, D)$$

(5d) $$\lambda_k \geq 0 \ (k = 1,2, \ldots, K) \ ; \ \theta_\ell \geq 0 \quad (\ell = 1,2, \ldots, L) \ .$$

Thus, the problem is changed to one with $D + 1$ constraints but many variables. The new form of the dual master is:

(6a) $$\text{Max} \quad L = \sigma + \sum_d T_d \pi_d$$

(6b) $$\sigma + \sum_d T_d^k \pi_d \leq F^k \qquad (k = 1,2, \ldots, K)$$

(6c) $$\sum_d U_d^\ell \pi_d \leq G^\ell \qquad (\ell = 1,2, \ldots, L)$$

(6d) $$\sigma \ , \ \pi_d \ \text{unrestricted} \quad (d = 1,2, \ldots, D) \ .$$

σ and the $\{\pi_d\}$ will always be nonnegative in any case of interest, since weakening (5b), (5c) to inequalities will not change any optimal solution.

As is usual in decomposition, one does not need *all* extremal solutions and rays, to start with, but only the $D + 1$ necessary to form an initial feasible basis to (5). New candidates will be generated as needed by solving the subproblem and added during the course of the algorithm. At any stage, if the *current* optimal dual variables are $\left\{\sigma^o, \pi_d^o\right\}$, a finite candidate solution $k = *$ to the subproblem can be checked by testing dual constraint (6b). Thus, if:

(7) $$F^* - \sum \pi_d^o T_d^* < \sigma^o$$

then the values $\left\{F^*;1;T_1^*,T_2^*, \ldots, T_D^*\right\}$ should be adjoined as a new column to the basis of the master, and a new optimum found by using the simplex method.

The search for such candidates is carried out by attempting to maximize the form (7) in the constraints (1b), (1c). In the original notation this leads to the *associated subproblem*:

(8a) $$\text{Min} \quad E = v_N - v_1 - \sum_{ij \epsilon A} C_{ij}^o t_{ij}$$

(8b) $$v_j - v_i - t_{ij} \geq T_{ij}$$

(8c) $$t_{ij} \geq 0 \,, \qquad \forall \; ij \; \epsilon \; A$$

with *dual associated subproblem:*

(9a) $$\text{Max} \quad K = \sum_{ij \epsilon A} T_{ij} x_{ij}$$

(9b) $$\sum_j (x_{ji} - x_{ij}) = \begin{cases} -1 & i = 1 \\ +1 & i = N \\ 0 & \text{otherwise} \end{cases}$$

(9c) $$x_{ij} \geq C_{ij}^o \geq 0 \qquad \forall \; ij \; \epsilon \; A \,.$$

Here C_{ij}^o is the *current unit savings* for lengthening ij, taken from (7) as:

(10) $$C_{ij}^o = \begin{cases} \pi^o_{A^{-1}(ij)} & ij \; \epsilon \; A - A_o \\ 0 & ij \; \epsilon \; A_o \,. \end{cases}$$

In other words, C_{ij}^o is the current dual price associated with a divisible activities cohort, and is nonnegative, by a previous remark.

The reason for optimizing (8a) is that if Min $E = E^*$, and one tests:

(7') $$E^* < \sigma^o$$

then this optimal extremal solution $\left\{x_{ij}^{*}, t_{ij}^{*}\right\}$ is clearly a candidate for the master. But if the minimal solution does not satisfy (7'), then no new candidates exist, and the current master must be optimal. Note that any unbounded optimal solution to (8) will always satisfy (7').

3. THE NATURE OF UNBOUNDED SOLUTIONS

The associated subproblem (8), (9) is in cost-time problem form, except there is no limit on the savings possible for extending activities with large C_{ij} , i.e., the optimal value of E may be unbounded. In dual terms, the exogenous flow $Q = 1$ may be insufficient to "cover" several large lower bounds C_{ij} on flow in arcs which are in a parallel configuration, i.e., (9) may be infeasible.

Figure 2 clarifies the nature of unbounded solutions. Suppose the set of all nodes, N , is partitioned into a subset X containing node 1 , and the complementary set $N - X$ containing node N ; the set of all arcs with one extremity in X , the other in $N - X$, is called a *cut*. In Figure 2, we have an *oriented cut* X , which all arcs ij are in $(X,N - X)$, i.e., they pass "from left to right." Most of the cuts in actual critical path networks are so oriented.

Let $\left\{v_i^o, t_{ij}^o\right\}$ be an (8) feasible solution to Figure 2. Then it is clear that:

$$(11)\qquad \begin{array}{ll} v_i' = v_i^o & i \in X \\ v_i' = v_i^o + \theta & i \in N - X \\ t_{ij}' = t_{ij}^o + \theta & ij \in (X,N - X) \\ t_{ij}' = t_{ij}^o & \text{otherwise} \end{array}$$

is also feasible for all $0 \leq \theta \leq \infty$. Thus, an oriented cut represents a possible unbounded solution to (8). One can easily show that no other possible infinite rays exists, assuming that the usual ambiguity in the $\{v_i\}$ is resolved by, say, setting $v_1 = 0$. For example, reversing a single arc in Figure 2 leads immediately to an upper bound on θ .

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|cc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & t_{12} & t_{13} & t_{14} & t_{23} & t_{24} & t_{35} & t_{45} & & \\
\hline
x_{12} & -1 & +1 & & & & -1 & & & & & & & & 0 \\
x_{13} & -1 & & +1 & & & & -1 & & & & & & & 1 \\
x_{14} & -1 & & & +1 & & & & -1 & & & & & & 2 \\
x_{23} & & -1 & +1 & & & & & & -1 & & & & & 2 \\
x_{24} & & -1 & & +1 & & & & & & -1 & & & \geq & 3 \\
x_{34} & & & -1 & +1 & & & & & & & & & & 2 \\
x_{35} & & & -1 & & +1 & & & & & & -1 & & & 6 \\
x_{45} & & & & -1 & +1 & & & & & & & -1 & & 3 \\
\hline
y_1 & & & & & & 1 & 1 & 1 & & & & & = & T_1 \\
y_2 & & & & & & & & & 1 & 1 & 1 & 1 & & T_2 \\
\hline
\end{array}
$$

FIGURE 1: CONSTRAINT MATRIX AND REQUIREMENTS VECTOR FOR EXAMPLE OF SECTION 8

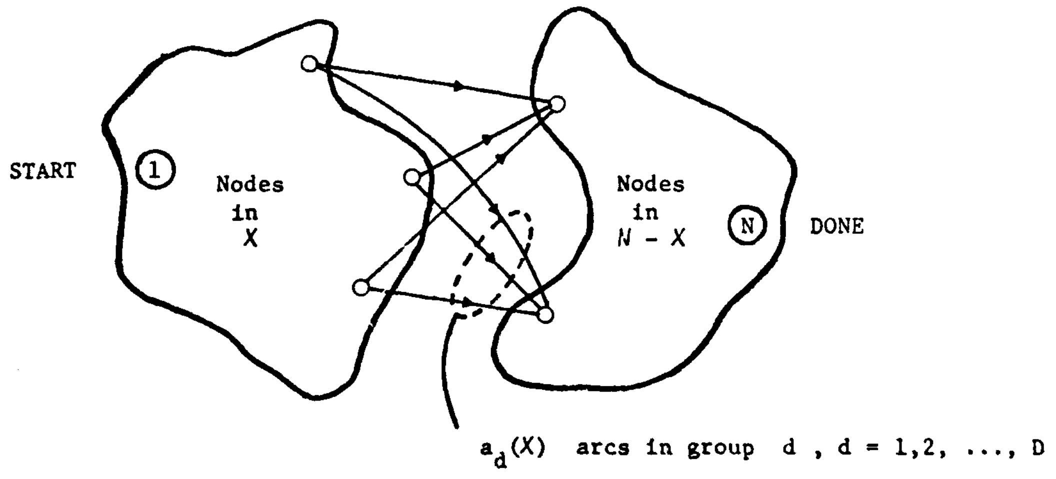


FIGURE 2: ORIENTED CUT OBTAINED FOR UNBOUNDED SOLUTIONS TO SUBPROBLEM

Let $a_d(X)$ be the number of arcs of divisible class d across oriented cut X. Unbounded solutions to (8) can be found by looking for infeasible solutions to (9b), (9c). For example, we could set the exogenous flow equal to Q, and solve a minimum flow problem; if $Q^* = \text{Min } Q > 1$, then (9b), (9c) is infeasible. However, by using the *min flow = max cut* theorem, it follows that an oriented cut X represents an unbounded solution to (8) iff:

$$\sum_d a_d(X)\pi_d^o > 1 . \tag{12}$$

An economic interpretation of (12) is that unbounded solutions arise when the total unit profitability for expanding an oriented cut is greater than the resulting "cost" of increasing the project duration by one time unit.

4. THE MASTER PROBLEM

The master problem needs at least D + 1 independent solutions to begin. For example, if an extremal solution k to (1b), (1c) (or even just a feasible solution) is known then the associated column in (5) is, transposed:

$$\left[E^k;1;T_1^k,T_2^k, \ldots, T_D^k\right] . \tag{13}$$

(the first position is for the objective function). An efficient bounded starting solution is given in Section 5.

However, in many problems, the rays associated with unbounded solutions play an important role. From the above discussion, the candidate column associated with an oriented cut X will be, transposed:

$$[1;0;a_1(X),a_2(X), \ldots, a_D(X)] . \tag{14}$$

Since these oriented cuts and their associated coefficients can be easily generated by various simple algorithms, it is advantageous to use as many of them as necessary in the initial basis; indeed, the optimal basis often consists of D rays and one finite solution. For small problems, one can even enumerate all oriented cuts and start with the "best" ones (largest sums $\sum a_d(X)$).

The resulting master program is solved by the simplex method, and then resolved as new columns are generated by the subproblem until finally a finite solution is generated which fails test (7'). It is advantageous to carry along an adjoined unit matrix, so that the current inverse of the basis can be used to update entering columns. Columns dropped from the basis tend not to reappear, but there is no guarantee of this.

5. AN EFFICIENT BOUNDED STARTING SOLUTION

It is inefficient to use the ordinary critical path solution to (1) (with all $T_d = 0$) as a starting solution, since even if $v_N - v_1$ is maintained at its minimal duration, F^o, there is almost always enough slack (float) off the critical path to "tuck in" some divisible time "for free." The following procedure is suggested as a good solution to use as the $k = 1$ column in the master. (It is assumed that the critical path problem can be solved repeatedly in an efficient manner, and that a ranked list of currently unallocated T_d is kept, and continuously updated.)

(15) (a) Set all $T_d = 0$, and solve (1) for its minimal solution $\left\{F^o; v_i^o; t_{ij}^o = 0\right\}$, with *earliest* possible event times. The resulting *tight* activities constitute an *early tree* with a *trunk* (critical path) from node 1 to N, and *branches* to all other nodes. For all nodes on the trunk, $v_i^1 = v_i^o$. $(F^1 = F^o)$, and for all arcs on the critical path $t_{ij}^1 = 0$.

(b) If ij is a *cotree arc* belonging to a currently unallocated class $A^{-1}(ij) = d$, let $t_{ij}^o \Leftarrow \text{Min}\left[v_j^o - v_i^o - T_{ij} - t_{ij}^o\text{ ; currently unallocated } T_d\right]$. (Arcs may be taken in any order.)

(c) Keeping the t_{ij}^o of (b) fixed, resolve for the *late tree* (latest possible event times). Repeat (b) for any cotree arcs in a currently unallocated cohort.

(d) Keeping the current t_{ij}^o fixed, find the earliest $\left\{v_i^E\right\}$ and latest $\left\{v_i^L\right\}$ event times from two passes of the critical path algorithm. For each currently unallocated class d, rank the arcs $ij \in A_d$ on the basis of largest available slack

$$s_{ij} = v_i^E - v_j^L - t_{ij}^o - T_{ij} .$$

(e) Select the class d with currently largest unallocated T_d *and* positive slack in some arc. For the $ij \in A_d$ with largest s_{ij}, let $t^o_{ij} \Leftarrow \text{Min}\,[s_{ij}\,;\ \text{currently unallocated } T_d]$.

(f) Repeat (d) finding the new event times until either:

(i) All $\{T_d\}$ are allocated; the solution is trivial.

or,

(ii) There exist unallocated divisible classes, and all arcs in these classes have zero slack, i.e., there are multiple critical paths of tight arcs. Set $t^1_{ij} = t^o_{ij}$, and the v^1_i to either of the values obtained during the last pass (d) - (f) during which no allocations occurred.

There are, of course, many other good heuristic procedures which can be used. The advantage of the above method is that it requires only two critical path solutions for (a) and (b), which often takes care of much of the "tuckable" t_{ij}, and then only two solutions plus some ranking per pass (d) - (f). Since (e) either increases the number of tight arcs, or completes the allocation of some class, the procedure is finite.

We believe this process gives a reasonably good starting solution. However, notice that it need not be near the final optimal allocation (which may require large allocations on one arc), nor is it optimal in the sense of allocating the maximal possible sum of all t_{ij}, since fractional solutions are not allowed.

Once the partial network of tight arcs obtained in this solution has been identified, it is trivial to devise algorithms to generate the D oriented cuts needed for the rest of the initial basis.

6. SOLVING THE SUBPROBLEM

Some remarks on solving (8), (9) are in order. If a cost-time (CPM) computer code [3], [7] is available, then (8) could be solved with functional

(8a') $$\text{Min} \quad E = Q(v_N - v_1) - \sum_{ij \varepsilon A} C_{ij} t_{ij} ,$$

and

(15) $$(\text{crash duration})_{ij} = T_{ij} ,$$

(16) $$(\text{normal duration})_{ij} = T_{ij} + M_{ij} ,$$

where M_{ij} is some large but finite number, say $M_{ij} = M$ for all arcs. To find infinite rays, it would be best if M were set larger than any number to which it were compared during the course of calculation; and it was possible to run the algorithm backwards, i.e., with decreasing Q. An infeasible solution would be detected at the breakpoint where Q attained unity, and the cut would be the set of arcs with $t_{ij} = M$.

An alternative method which uses the CPM code in the usual manner is to set:

(17) $$M_{ij} = T_{A^{-1}(ij)} \qquad \forall \ ij \ A - A_o .$$

This will give only finite solutions to the master problem. If there are an infinite number of optimal solutions to $Q = 1$ (i.e., there are two breakpoints $Q = \alpha$ and $Q = \beta$, with $\alpha < 1 < \beta$), then both extreme solutions should be furnished to the master for optimal mixing.

Another possibility is to work with the dual (9), using an *out-of-kilter* code [3]. Figure 3 shows the complementary slackness diagram which is appropriate to the dual flow problem, assuming the code is of minimizing type. All other things being equal, it is desirable to have t_{ij} as large as possible, so that a feasible solution $x_{ij} = C_{ij}$ should be interpreted as being a corner solution (i.e., the relation (8b) is tight). If the dual is infeasible, the labelling subroutine of the out-of-kilter method will stop with some nodes labelled, and the remainder not; these labels then define the desired oriented cut X .

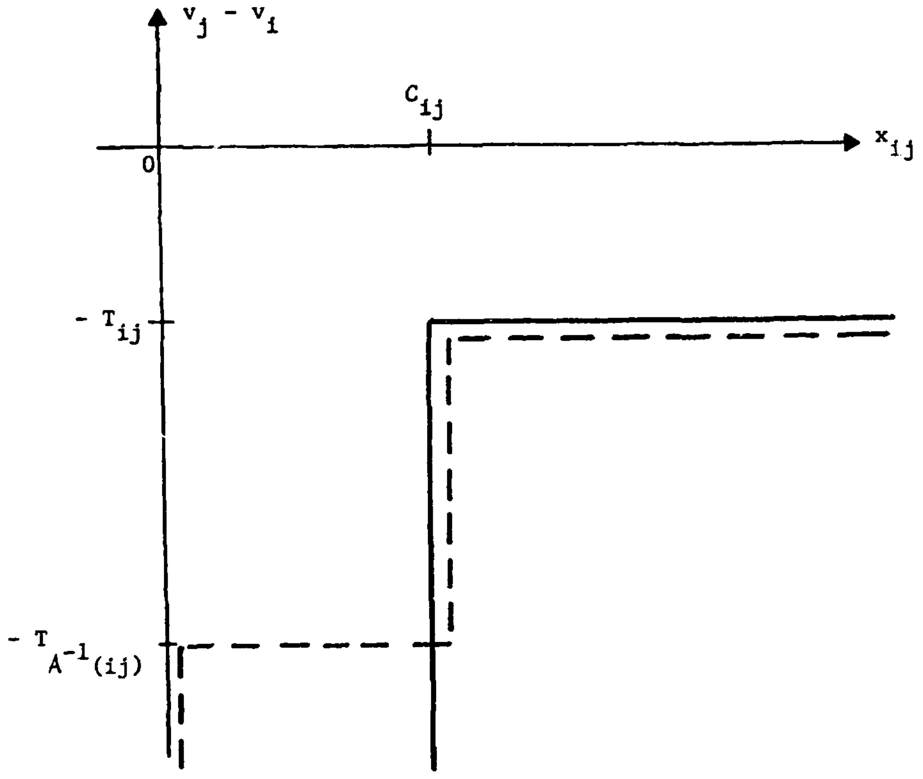
Alternatively, one can place two arcs in parallel, with zero lower flow bounds, and use parameters:

(18)

	Unit Cost	Upper Flow Bound
Arc (ij)'	$-T_{A^{-1}(ij)}$	C_{ij}
Arc (ij)"	$-T_{ij}$	∞

giving the composite complementary slackness diagram shown by the "staircase" dotted lines in Figure 3. This procedure gives only finite solutions to the master. Again, if solutions on the "riser" of Figure 3 are obtained, both endpoints should be given to the master.

Most computer codes require integer C_{ij} , whereas ours may be fractional. However, there always exists some large integer J such that $C'_{ij} = JC_{ij}$ is integer for all ij . Hence, an equivalent problem can be run with integer C_{ij} , and flow and breakpoint $Q = J$. The resulting flows and functionals will all be too large by the same factor; however, the new plan or cut will still be the same. Section 9 discusses a choice of J which need not be changed at every pass of the subproblem.



—— Infeasible Solutions Feasible

– – – Feasible Solutions Only

FIGURE 3: COMPLEMENTARY SLACKNESS DIAGRAM FOR SOLVING SUBPROBLEM WITH MINIMAL COST OUT-OF-KILTER ALGORITHM

7. THE DIVISIBLE ACTIVITIES ALGORITHM

Figure 4 summarizes, in flow chart form, the final decomposition algorithm proposed for solving the divisible activities problem. Here t is the index of iterations, $t = 0,1,2, \ldots$ Notice that the problem is always feasible (assuming the ordinary critical path problem is) and bounded.

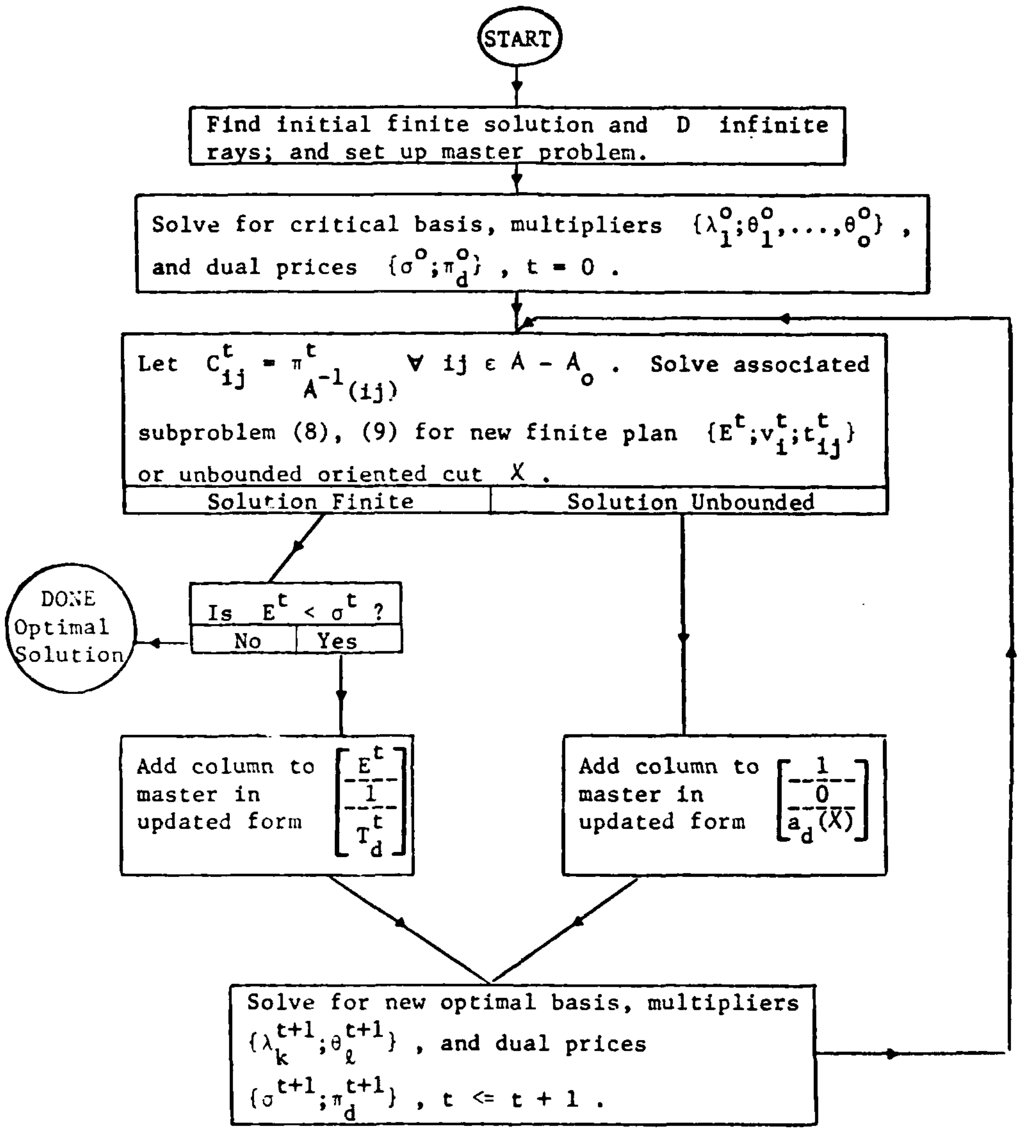

START
Find initial finite solution and D infinite rays; and set up master problem.
Solve for critical basis, multipliers $\{\lambda_1^o;\theta_1^o,\ldots,\theta_o^o\}$, and dual prices $\{\sigma^o;\pi_d^o\}$, $t = 0$.
Let $C_{ij}^t = \pi^t_{A^{-1}(ij)}$ $\forall\ ij \in A - A_o$. Solve associated subproblem (8), (9) for new finite plan $\{E^t;v_i^t;t_{ij}^t\}$ or unbounded oriented cut X.
Solution Finite
Solution Unbounded
Is $E^t < \sigma^t$?
No
Yes
DONE Optimal Solution
Add column to master in updated form $\begin{bmatrix} E^t \\ 1 \\ T_d^t \end{bmatrix}$
Add column to master in updated form $\begin{bmatrix} 1 \\ 0 \\ a_d(X) \end{bmatrix}$
Solve for new optimal basis, multipliers $\{\lambda_k^{t+1};\theta_\ell^{t+1}\}$, and dual prices $\{\sigma^{t+1};\pi_d^{t+1}\}$, $t \Leftarrow t + 1$.

FIGURE 4: FLOW DIAGRAM OF DIVISIBLE ACTIVITIES ALGORITHM

8. EXAMPLES OF DIVISIBLE ACTIVITY ALLOCATION

Figure 5 shows the network corresponding to the matrix of Figure 1. The numbers on the nodes are the node indices; the numbers on the arcs represent the $\{T_{ij}\}$. Figure 6 shows the early tree solution when $T_1 = T_2 = 0$ and corresponding event times on the nodes, the late solution also has $F = 8$, but $v_4^L = 5$.

Henceforth, the node numbers will represent the $\{v_i\}$, and the arc numbers will be the $\{t_{ij}\}$. Tight arcs will be solid and slack arcs dotted. Figure 7 shows the initial bounded solution obtained when

$$T_1 \geq T_1^1 = 3 \; ; \; T_2 \geq T_2^1 = 2$$

and the algorithm in (15) is used. All arcs are tight.

1. For the first example, suppose $T_1 = 4$ and $T_3 = 3$, and choose as cuts

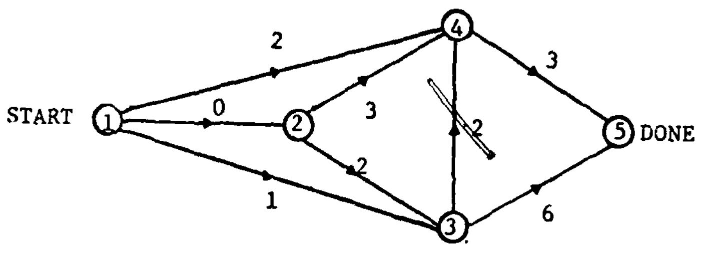
$$X^1 = \{1\} \; ; \; X^2 = \{1,2,3,4\} \; .$$

The initial form of the master tableau is:

	λ_1	θ_1	θ_2				
	8	1	1	0	0	0	0
σ	1	0	0	1	1	0	0
π_1	3	3	0	4	0	1	0
π_2	2	0	2	3	0	0	1

where the top row is the functional F , and a unit matrix has been adjoined to provide the basis inverse.

The usual reduction methods give the reduced tableau:



$A_o = \{(3,4)\}$

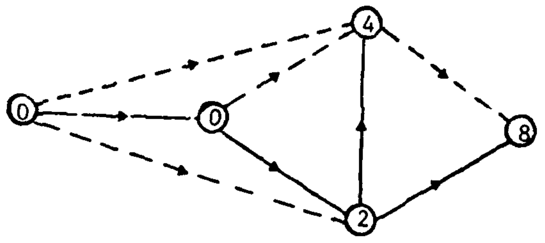
$A_1 = \{(1,2),(1,3),(1,4)\}$

$A_2 = \{(2,3),(2,4),(3,5),(4,5)\}$

Node numbers - indices

Arc numbers - T_{ij}

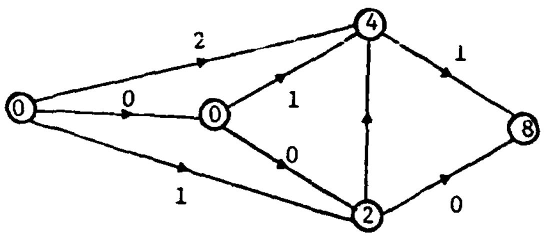
FIGURE 5: ORIGINAL NETWORK EXAMPLE



Node numbers - v_i

Tight arcs - solid

FIGURE 6: EARLY TREE CRITICAL PATH SOLUTION



$T_1 > T_1^1 = 3$

$T_2 > T_2^1 = 2$

Arc numbers - t_{ij}

FIGURE 7: INITIAL BOUNDED SOLUTION (λ_1) FOR FIRST EXAMPLE, WITH UNALLOCATED T_1 , T_2

λ_1	θ_1	θ_2				
0	0	0	$-8\frac{5}{6}$	-6	$-\frac{1}{3}$	$-\frac{1}{2}$
1	0	0	1	1	0	0
0	1	0	1/3	-1	1/3	0
0	0	1	1/2	-1	0	1/2

from which we conclude $\lambda_1 = 1$, $\theta_1 = 1/3$, $\theta_2 = 1/2$; $\sigma = 6$, $\pi_1 = 1/3$, $\pi_2 = 1/2$ and $F = 8\ 5/6$. The resulting allocation after the first iteration is shown in Figure 8.

Next a flow subproblem with $C_{ij} = 1/3\ (ij \in A_1)$ and $C_{ij} = 1/2\ (ij \in A_2)$ is attempted, and an infeasible solution obtained. A flow $Q = 1\ 2/3$ will "cover" all the arcs, but attempts to reduce Q to 1 lead to a max cut $X^3 = \{1,2\}$. Consequently, a column θ_3 is added to the master with values $[1,0,2,2]^{\text{transpose}}$, after updating in the usual way. θ_1 drops from the basis and the new optimal solution is $\lambda_1 = 1$, $\theta_2 = 0^+$, $\theta_3 = 1/2$; $\sigma = 7$, $\pi_1 = 0$, $\pi_2 = 1/2$ and $F = 8\ 1/2$. The second allocation is shown in Figure 9.

When the flow subproblem is solved, a finite flow solution is obtained, with $x_{12} = x_{23} = 1$, $x_{34} = x_{35} = x_{45} = 1/2$ and $K = 7\ 1/2$. The corresponding primal would be $t_{12} = 0$, $t_{13} = 1$, $t_{14} = 2$; $t_{23} = 0$, $t_{24} = 1$, $t_{35} = \theta$, $t_{45} = 1 + \theta$ $(0 \leq \theta \leq \infty)$, but we know already from duality that $E = 7\ 1/2\ (\forall \theta)$, test (7') fails, and this is not a candidate solution. Hence, Figure 9 is the optimal allocation to the first example.

II. As a second example, take $T_1 = 5$ and $T_2 = 3$. The same sequence of tableaux is obtained except that θ_3 displaces θ_2 , and $\lambda_1 = 1$, $\theta_1 = 1/3$, $\theta_3 = 1/2$; $\sigma = 6\ 2/3$, $\pi_1 = 1/2$, $\pi_2 = 6$ (Figure 10). The optimal flow dual subproblem has $x_{35} = x_{45} = 1/2$ and $K = 6\ 1/3$. Since test (7') is satisfied,

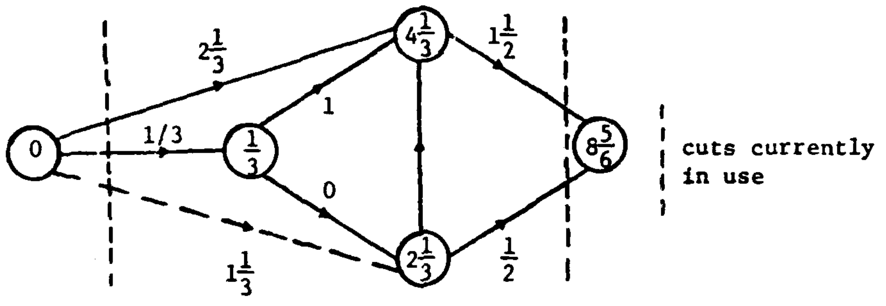


FIGURE 8: INITIAL MASTER SOLUTION FOR FIRST EXAMPLE, WITH ALL DIVISIBLE TIME ALLOCATED

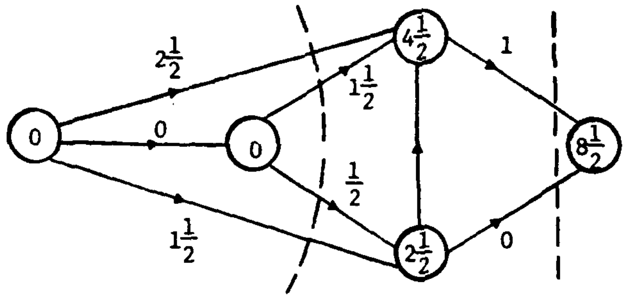


FIGURE 9: SECOND (AND OPTIMAL) SOLUTION TO FIRST EXAMPLE

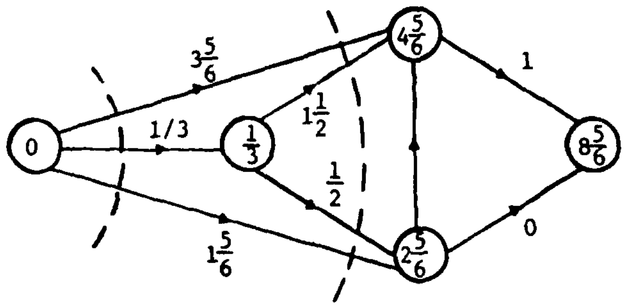


FIGURE 10: SECOND MASTER SOLUTION TO SECOND EXAMPLE

we must find the new finite primal subproblem (λ_2) , which is shown in Figure 11. Note that fractional extremal solutions to subproblems can arise (with fractional C_{ij}) , and that (3,4) is slack for the first time. A new column $[8\ 1/2;1;5,3]^{\text{transpose}}$, is added to the master after updating, and λ_2 drives out (say) λ_1 , leaving $\theta_1 = \theta_3$ at zero level in the basis. $\sigma = 6\ 1/3$, and $\pi_1 = 1/3$, $\pi_2 = 1/6$ as before, and this time the test fails as an equality. In other words, the bounded extremal solution $\lambda_2 = 1$ is *by itself* the optimal solution (another example of this occurs in the example given in [4], where $D = 1$) . The additional unit of T_1 is handled without increasing the project duration.

III. As the third example, let $T_1 = 4$ and $T_2 = 2$. By inspection of λ_1 we see that all T_2 is taken care of, and the cheapest way to handle the remaining unit of T_1 is to put as much of it in parallel as possible, i.e., to divide it into thirds. The optimal solution is shown in Figure 12, where $\lambda_1 = 1$, $\theta_1 = 1/3$, $\theta_3 = 0^+$; $\sigma = 6\ 2/3$, $\pi_1 = 1/3$, $\pi_2 = 1/6$.

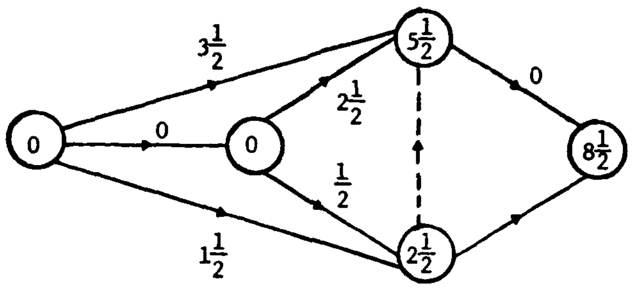


FIGURE 11: NEW FINITE EXTREMAL SOLUTION (λ_2) FOR SECOND EXAMPLE (ALSO OPTIMAL)

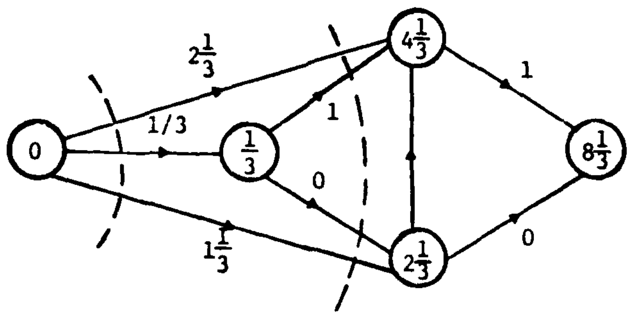


FIGURE 12: OPTIMAL SOLUTION TO THIRD EXAMPLE

9. SOLUTION MODULARITY

The reason that fractional solutions arise is because, when a_d parallel critical arcs are available for allocation, it is optimal to divide each hour of unallocated time into the fractions $1/a_d$, and spread the work uniformly over the critical paths. Following the arguments advanced in [3], we see that if $D = 1$, then only solutions in multiples of $1/1, 1/2, \ldots, 1/A_1$ could be obtained. With $D > 1$, it follows that, if

$$A_* = \max_{d=1,2,\ldots,D} A_d , \tag{19}$$

then multiplication of the original duration data by $(A_*)!$ must lead to a solution in integers. In the terminology of [6], the problem is A_* -*modular*.

This result is primarily of theoretical interest; however, it does indicate that $J = (A_*)!$ is a suitable choice when using an integer code as described in Section 6.

10. MOVABLE ACTIVITIES

A different type of problem arises when the constraints (1c) (1d) are replaced by

(20)
$$\sum_{ij \varepsilon A_d} t_{ij} = T_d \qquad d = 1,2, \ldots, D$$
$$t_{ij} = 0 \ \textit{or} \ T_{A^{-1}(ij)} \qquad \forall \ ij .$$

In this case, we consider that the work of type d is not *divisible*, but is only *movable*, i.e., it may be performed only at one of several possible locations in the project network. This problem is one of *binary programming*, since one can introduce binary variables, δ_{ij}, with

(21)
$$t_{ij} = \delta_{ij} \cdot T_{A^{-1}(ij)} .$$

Similar formulations arise if we consider that the job content consists of indivisible modules of work, $\left\{T_{d1}, T_{d2}, \ldots, T_{dM_d}\right\}$, such that

$$\sum_{m=1}^{M_d} T_{dm} = T_d \qquad (d = 1,2, \ldots, D) .$$

Conceptually, one separates arcs in group d into M_d arcs in series, and treats each module as a separate movable group. A special example of interest is when these modules are of the same length, say one unit of time; this would be the problem of *divisible activities with integer solutions*.

There are also mixed-integer formulations possible, but these will be seen to be a special case of the following approach. They are also of less practical interest.

11. SEPARATION IN A BRANCH-AND-BOUND METHOD

The method proposed to solve the problem of movable activities is a branch-and-bound procedure, modified to take account of the special structure of the problem. Readers not familiar with this approach are referred to Lawler and Wood [8], or Bertier and Roy [1].

The first element to be considered is the *branching (arbitration)*, i.e., the separation of the solution space into subspaces where certain of the binary variables are fixed once and for all. This separation is conveniently represented by an *arborescence*, where each node represents a possible partition of the solution space, and each arc shows the branching between a given arbitration and the further possible partitions.

The natural integer programming separation would come about by considering some (possibly adaptive) ordering $(i_1j_1, i_2j_2, i_3j_3, \ldots)$ of the δ_{ij}, and making the arbitration on whether $\delta_{ij} = 0$ or 1. This leads to the arborescence shown in Figure 13. However, this method tends to postpone the labor involved in making and evaluating the decision as to where the work T_d should be performed, since the bound used for $\delta_{ij} = 0$ always leads to further exploration of that "branch," until some $\delta_{ij} = 1$ is selected.

A more compact representation is the *task-oriented* arbitration shown in Figure 14, in which a (possibly adaptive) ordering of the movable tasks is made, and each separation represents a definite allocation of T_d to one of A_d locations.

We call an *arbitration of order* k a problem in which exactly k of the δ_{ij} have been fixed at 0 or 1. The result can be considered a new problem with

$$(22) \qquad T'_{ij} = \begin{cases} T_{ij} + T_{A^{-1}(ij)} & ij \text{ arbitrated and } \delta_{ij} = 1 \\ T_{ij} & \text{otherwise;} \end{cases}$$

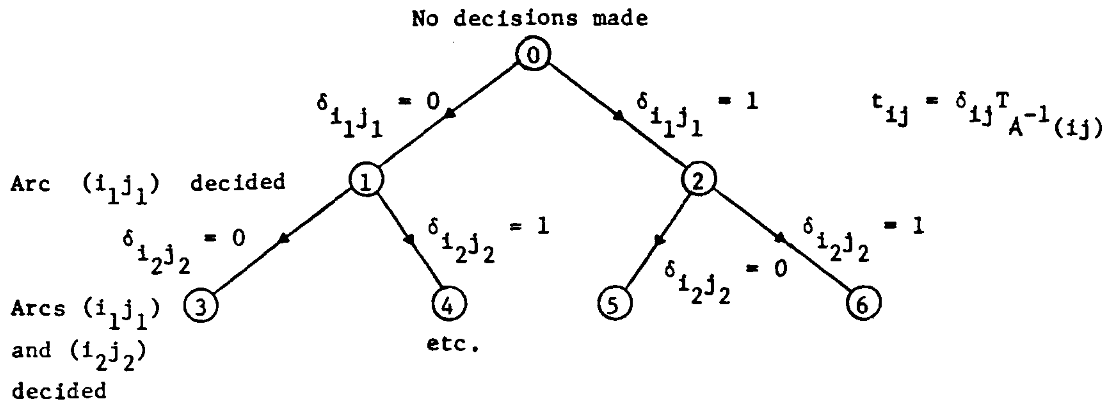


FIGURE 13: ARC-ORIENTED SEPARATION OF MOVABLE ACTIVITY SOLUTION SPACE

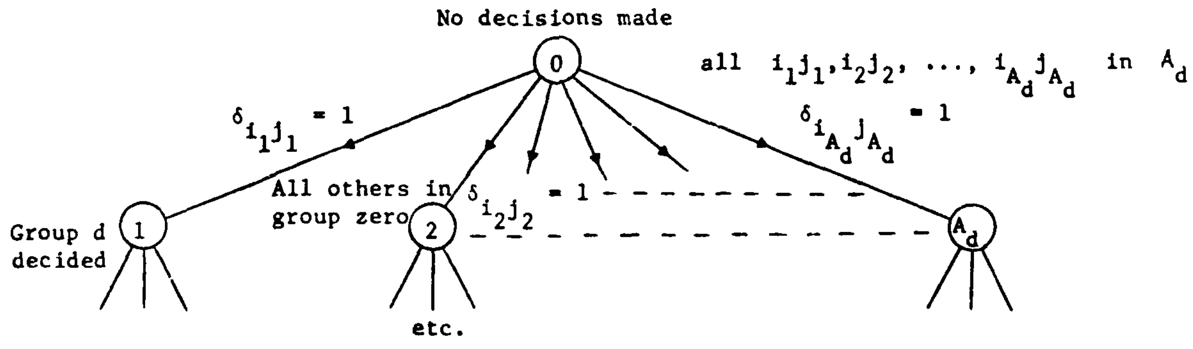


FIGURE 14: TASK-ORIENTED SEPARATION OF MOVABLE ACTIVITY SOLUTION SPACE

the t_{ij} are then eliminated from the arbitrated arcs, giving a reduced problem in $A - A_o - k$ free t_{ij} .

If we use the task oriented separation of Figure 4, we see that the arbitration jumps from order 0 to order A_d from the first to the second level, leaving a new problem with one group of t_{ij} eliminated, and only $D - 1$ more levels to go.

12. EVALUATION IN A BRANCH-AND-BOUND METHOD

The other element needed in the algorithm is an efficient method of bounding the optimal value of the functional, F, from below in any partially arbitrated problem. Thus, for any arbitration a in the arborescence of Figure 14 ($a = 0,1, \ldots, A-A_o$), we know that there will be a best value for $F(a) = F^*(a)$ in the reduced problem governed by (22) for the already arbitrated values of δ_{ij}. What we seek is an *evaluation*, $\underline{E(a)}$, of this optimum satisfying

$$(23) \qquad \underline{E(a)} \leq F^*(a) \leq \overline{E(a)} ,$$

one which can be found *without* calculating all of complete solutions *(terminal nodes)* which are *descendants* of node a. (It also turns out to be easy and worthwhile to find an upper bound, $\overline{E(a)}$, at the same time, as indicated.) These bounds are then used to *guide* the exploration of the arborescence in an efficient manner, thus avoiding an exhaustive check of all possible 2^{A-A_o} solutions to the primitive problem (20).

For example, at the parent node $a = 0$ of Figure 14, we have the original movable activities problem. Call $F^{\#}$ the optimal value of F for the corresponding divisible activities problem (1), and F^0 the critical path solution of (1) with all $t_{ij} = 0$; it follows that

$$(24) \qquad \underline{E(0)} = F^{\#} \leq F^*(0) \leq F^0 + \sum_{d=1}^{D} T_d = \overline{E(0)}$$

are good evaluators in the sense that there are networks for which one (or both) of the evaluators will be attained by F^*.

Similarly, define $F^{\#}(a)$ and $F^0(a)$ as the optimal values of the divisible activities and ordinary critical-path functionals, respectively, of the problem of arbitration, reduced by applying (22). Then

$$(25) \qquad \underline{E(a)} = F^{\#}(a) \leq F^{*}(a) \leq F^{0}(a) + \sum_{\substack{\text{unarbitrated}\\ \text{groups } d}} T_d = \overline{E(a)}$$

are also good evaluators, in the same sense. (25) also has the desirable property that at a terminal node of Figure 14 (all arcs arbitrated), the bounds coincide, and $\underline{E} = F^{*} = \bar{E}$.

Unfortunately, using the divisible-activities algorithm is very inefficient. Not only may several simplex iterations be necessary to solve for a single $F^{\#}(a)$ but columns obtained in this solution cannot be used for another evaluation $F^{\#}(a')$, since they may not be feasible in the new subproblem of different arbitration.

However, we note that repeated computation of the ordinary critical path solution can be easily done for any arbitration, finding not only F^{0} , but both the early $\{v_i^E\}$ and late $\{v_i^L\}$ event times for all nodes. This observation leads to a weaker, but much rapid determination of a bound, $\underline{E(a)}$.

13. AN EFFICIENT LOWER BOUND

As in the algorithm of Section 5, define the *slack* of ij for any ordinary critical path solution to be

$$s_{ij} = v_i^E - v_j^L - T_{ij} \, . \tag{26}$$

Now suppose that there was only one group, A_1, of arcs left to arbitrate. Defining

$$S_d = \max_{ij \in A_d} (s_{ij}) \, , \tag{27}$$

we see that an *exact* solution of the one group problem is

$$\underline{E} = F^0 + \max(0, T_1 - S_1) = F^* \, ; \tag{28}$$

in other words, the remaining task is allocated to the largest slack location (or possibly to one of several locations, if $S_1 > T_1$) .

When there is more than one group to be arbitrated, the best possible situation is to pick that group first which gives the largest increase in F^0 ; this increase must be absorbed in any case, and it might then be possible to tuck in the remaining tasks in parallel gaps.

In other words, at arbitration a , solve for the two critical path solutions $\left\{v_i^E(a)\right\}$, $\left\{v_i^E(a)\right\}$ and the associated duration $F^0(a) = v_N^E(a) - v_1^E(a) = v_N^L(a) - v_1^L(a)$, using the appropriate T_{ij} . Then

$$\underline{E(a)} = F^0(a) + \max \left[0 \, , \max_{\substack{\text{unarbitrated} \\ \text{groups } d}} (T_d - S_d(a)) \right] \tag{29}$$

is an appropriate evaluator.

Note that if $S_d(a) \geq T_d$ for all remaining groups, we *cannot* necessarily conclude that the problem is over, since the largest slacks for each group may not be simultaneously available.

14. IMPROVING THE UPPER BOUND

The upper evaluator, $\overline{E(a)}$, given in (25) can easily be improved by, say, completing the arbitration a in any arbitrary manner, and taking the resulting terminal F^0 .

A simpler method, which does not require a third critical path solution, is to use the S_d found in (27). Suppose we pick the group to allocate next which gives the smallest increase $T_d - S_d$. Then, in the worst case, all of remaining slacks might have decreased to value zero, and all remaining unarbitrated jobs would have to be added in series to the critical path. Since we can pick this first group judiciously

$$(30) \qquad \overline{E(a)} = F^0(a) + \sum_{\substack{\text{unarbitrated} \\ \text{groups } d}} T_d - \max_{\substack{\text{unarbitrated} \\ \text{groups } d}} \min\,[S_d(a), T_d] \; .$$

One can improve this bound by looking ahead several steps, but it is probably not worth the extra effort.

15. SELECTING THE ARBITRATION

As is usual in branch-and-bound algorithms where the bounding procedure does not lose efficiency when skipping among the different candidates in the arborescence, we recommend that *the arbitration with smallest* $\underline{E(a)}$ *be separated next* during the algorithm.

If, at any stage, we discover $\overline{E(a_1)} < \underline{E(a_2)}$ for two candidate arbitrations, we can *truncate* the arborescence at node a_2, i.e., discontinue further exploration in descendant nodes. This illustrates possible value of carrying upper bounds.

Selecting the next movable task to be arbitrated should be done adaptively, and not in a fixed, predetermined order. Limited computational experience suggests that the following rule is quite efficient:

(31) Select as the next movable task to be separated from arbitration a that unarbitrated task d which maximizes $T_d - S_d(a)$. If there is a tie (or if all $T_d - S_d(a)$ are zero), select the task d with largest T_d.

In other words, it seems efficient to decide early where to put the jobs with the most excess duration; or, if there is a tie, to decide the job with largest duration.

Furthermore, we note when the separation occurs, we actually know in advance what the values of $F^0(a + 1)$ will be, from the value of T_d of the $\{s_{ij}\}$ belonging to this group. It is then efficient to carry out the evaluation of this separation *in order of decreasing* S_{ij} $(ij \in A_d)$. Often the upper bound from one placement of T_d in a large slack location will truncate the further exploration of the arborescence from another location with small slack, thus saving some computation. This observation is particularly true towards the bottom of the arborescence, when the spread between upper and lower evaluators is small.

16. THE MOVABLE ACTIVITIES ALGORITHM

Figure 15 shows the flow chart of the final algorithm. Several explanatory comments are perhaps in order.

At each stage of the algorithm, a candidate list, $\mathcal{L}$, of arbitrations (nodes) is kept; these are merely partially arbitrated solutions which have not been further explored. If one candidate is completely dominated by another, then it is eliminated from $\mathcal{L}$, and the arborescence is truncated, as described above.

Since the next arbitration in $\mathcal{L}$ to be elaborated further is the one of *current smallest* $\underline{E}$, the procedure may jump between nodes of different order, and the resulting sequence will not follow the prior numbering of the nodes. This rule requires more computer storage than other methods, such as "backtracking," but it does enable one to deduce that an optimal solution has been found, whenever an arbitration of maximal order is reached [1], [8] (assuming all ties in selecting $\underline{E}$ have been resolved).

Various other heuristics, perhaps utilizing the upper evaluator and/or the level of arbitration, are of course possible.

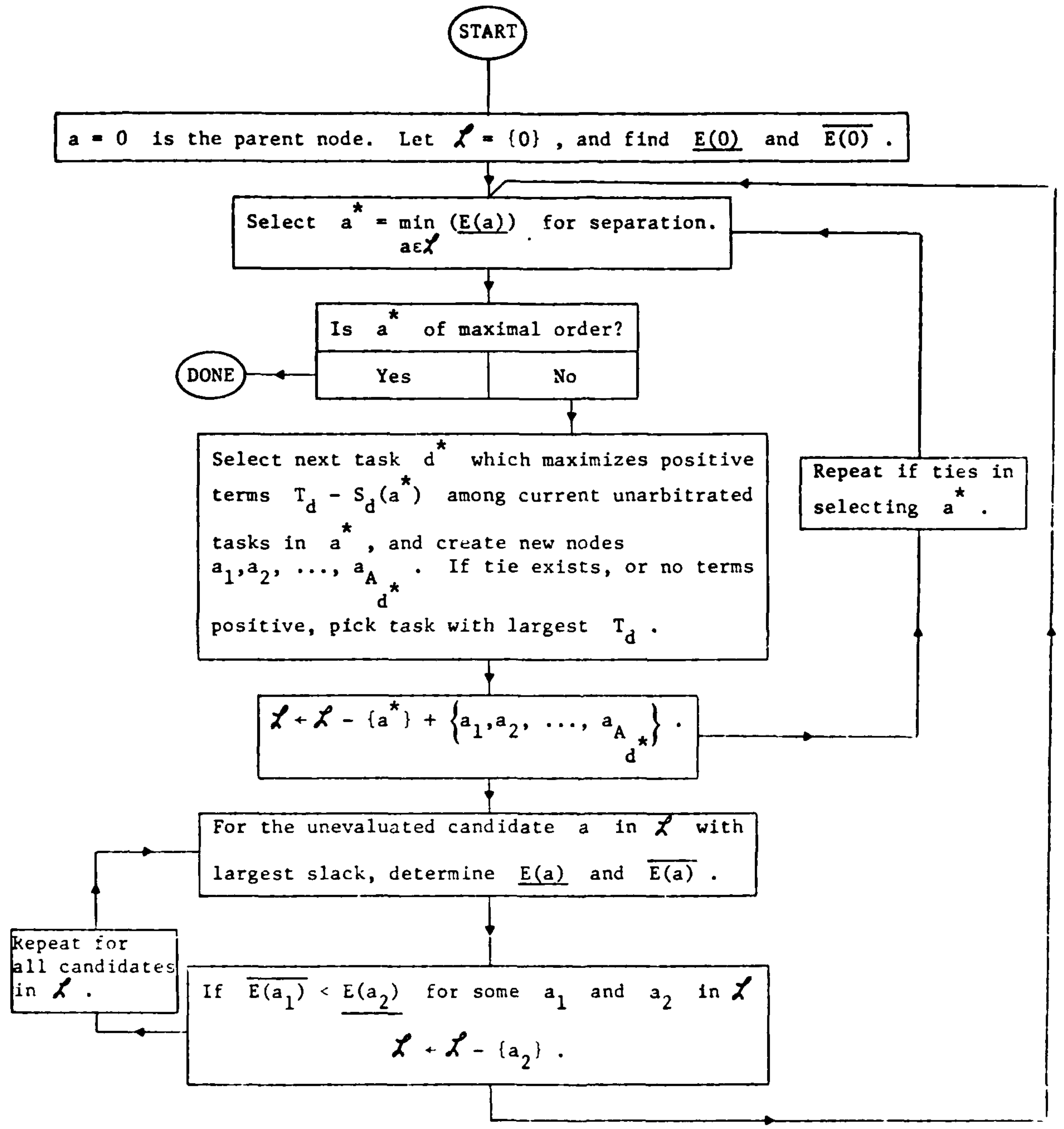

START
a = 0 is the parent node. Let 𝒳 = {0} , and find $\underline{E(0)}$ and $\overline{E(0)}$.
Select $a^* = \min_{a\varepsilon\mathcal{X}} (\underline{E(a)})$ for separation.
Is a^* of maximal order?
Yes
No
DONE
Select next task d^* which maximizes positive terms $T_d - S_d(a^*)$ among current unarbitrated tasks in a^* , and create new nodes $a_1, a_2, \ldots, a_{A_{d^*}}$. If tie exists, or no terms positive, pick task with largest T_d .
Repeat if ties in selecting a^* .
$\mathcal{X} \leftarrow \mathcal{X} - \{a^*\} + \{a_1, a_2, \ldots, a_{A_{d^*}}\}$.
For the unevaluated candidate a in 𝒳 with largest slack, determine $\underline{E(a)}$ and $\overline{E(a)}$.
Repeat for all candidates in 𝒳 .
If $\overline{E(a_1)} < \underline{E(a_2)}$ for some a_1 and a_2 in 𝒳
$\mathcal{X} \leftarrow \mathcal{X} - \{a_2\}$.

FIGURE 15: FLOW DIAGRAM OF MOVABLE ACTIVITIES ALGORITHM

17. EXAMPLE

Consider Figure 16, which shows the T_{ij} for each arc. Suppose there are four movable groups:

$$\begin{aligned}
A_1 &= \{(1,2),(1,3),(1,4),(5,8),(6,8),(7,8)\} \;; & T_1 &= 5 \\
A_2 &= \{(2,3),(3,4),(5,6),(7,6)\} \;; & T_2 &= 6 \\
A_3 &= \{(2,5),(3,6),(4,7)\} \;; & T_3 &= 4 \\
A_4 &= \{(3,5),(4,6)\} \;; & T_4 &= 8 \\
& & \sum T_d &= 23
\end{aligned}$$

From the ordinary critical path solution, Figure 17, we find that the slacks are (same order as above).

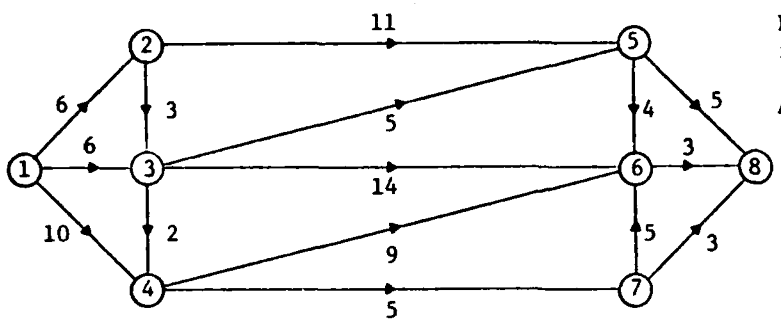
	$T_d - S_d$
$\{s_{ij} \mid ij \in A_1\} = \{0,0,2,4,0,1\}$	1
$\{s_{ij} \mid ij \in A_2\} = \{0,1,2,4\}$	2
$\{s_{ij} \mid ij \in A_3\} = \{2,0,1\}$	2
$\{s_{ij} \mid ij \in A_4\} = \{5,3\}$	3

Thus for the parent node, $a = 0$, we conclude

$$26 + 3 = 29 \le T^* \le 26 + 23 - 5 = 44 \; .$$

Since $T_d - S_d$ is largest for group 4, we arbitrate this first, $a = 1$ selects $(3,5)$, $a = 2$ selects $(4,6)$. Note that we have immediately the preliminary estimates $\underline{E(1)} = 29$, $\underline{E(2)} = 31$. Following our previous rule, we evaluate node 2 first, obtaining

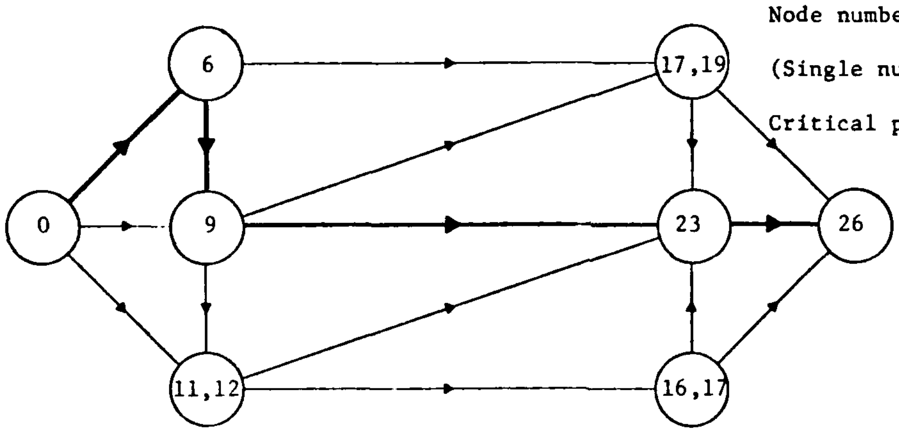
$$\underline{E(2)} = 31 \;, \; \overline{E(2)} = 29 + 15 - 6 = 38$$



Node numbers are indices

Arc numbers are $\{T_{ij}\}$

FIGURE 16: MOVABLE ACTIVITIES EXAMPLE

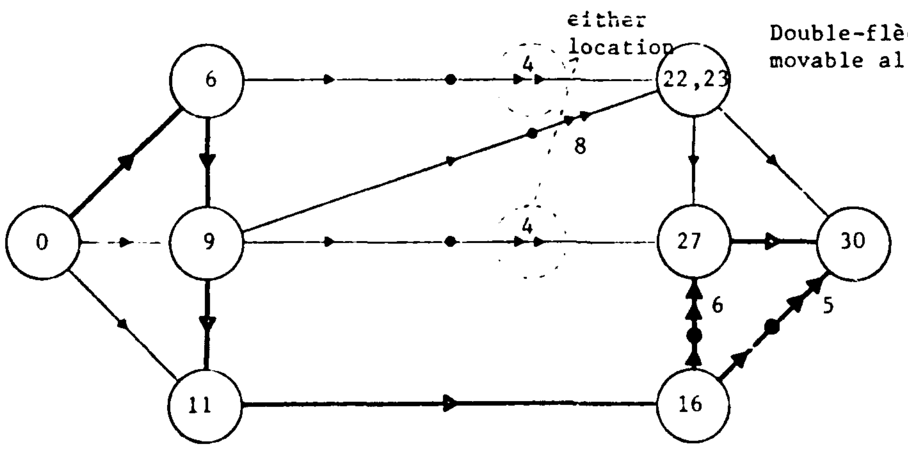


Node numbers are $\left\{v_i^E, v_i^L\right\}$

(Single number if identical)

Critical path is heavy line

FIGURE 17: ORDINARY CRITICAL PATH SOLUTION



Double-flèche arcs are movable allocations

FIGURE 18: OPTIMAL SOLUTION

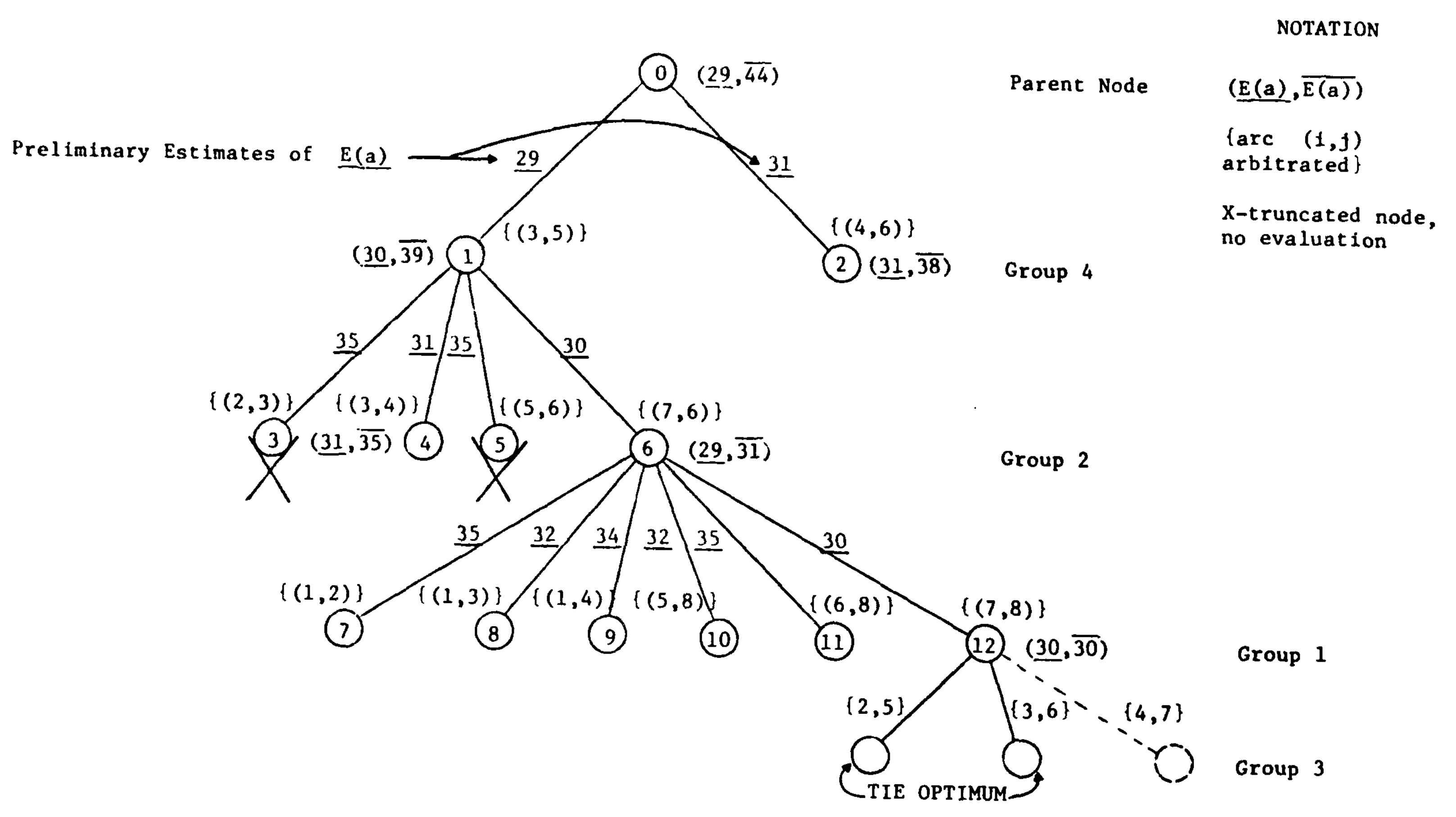


FIGURE 19: ARBORESCENCE FOR MOVABLE ACTIVITIES EXAMPLE

from the slacks {0,3,1,9,0,6} , {0,0,7,7} , and {7,5,6} of groups numbers one, two, and three respectively.

For node 1 , we obtain slacks {0,3,5,2,0,4} , {0,4,0,5} , and {5,3,4} for the same slacks, giving evaluations

$$\underline{E(1)} = 29 + 1 = 30 \ , \ \overline{E(1)} = 29 + 15 - 5 = 39 \ .$$

Since node 1 has the lowest $\underline{E}$, we arbitrate that node on group 2 , for which $T_d - S_d(1)$ is largest, equal to 1 . This creates new nodes 3 , 4 , 5 , 6 corresponding to selecting (2,3) , (3,4) , (5,6) , or (7,6) , respectively. We can get estimates of $\underline{E}$ = 35 , 31 , 35 , 30 for each of these nodes from the last step, and so we carry out the evaluations in the order a = 6 , 4 , 3 , 5 or a = 6 , 4 , 5 , 3 .

For a = 6 , the slacks are {0,3,1,3,0,5} and {6,4,0} for the remaining groups numbers one and three, so that

$$\underline{E(6)} = 30 \quad \text{and} \quad \overline{E(6)} = 30 + 9 - 5 = 34 \ .$$

From this upper bound, we can immediately truncate nodes 3 *and* 5 and not evaluate them.

For a = 4 , the slacks are {0,3,7,4,1,0} and {7,5,0} , giving

$$\underline{E(4)} = 31 \quad \text{and} \quad \overline{E(4)} = 31 + 9 - 5 = 35 \ .$$

The list $\mathcal{L}$ of the algorithm now contains nodes 2 , 4 , and 6 of the arborescence. We arbitrate node 6 as the most promising, choosing group number one since it has largest T_d (no terms $T_d - S_d$ positive). The result is the creation of nodes 7 , 8 , 9 , 10 , 11 , 12 corresponding to choosing arcs (1,2) , (1,3) , (1,4) , (5,8) , (6,8) , and (7,8) , respectively. The preliminary estimates of $\underline{E}$ are 35 , 32 , 34 , 32 , 35 , and 30 , so that the evaluations are to be made in the order $a = 12, 8, 10, 9, 7, 11$.

For $a = 12$, the slacks in the reamining group number three are $\{6,4,0\}$, so that there are *two* locations $(2,5)$, $(3,6)$ in which the four units of work could go. Hence,

$$\underline{E(12)} = \overline{E(12)} = 30 .$$

And we are done, because of the look-ahead property of our evaluators, and the fact that no remaining $\underline{E(a)}$, $a \in \mathcal{X}$, has value 30.

Thus, after six (earliest and latest) critical path solutions, we have the optimal arbitrations

$$\{(3,5),(7,6),(7,8),(2,5)\} \quad \text{or} \quad \{(3,5),(7,6),(7,8),(3,6)\} .$$

18. EXTENSIONS

One obvious remark is that, if a movable activities algorithm is available, then one can use it (with integer modules) to give a good starting solution to the completely divisible activities problem. However, as mentioned previously, it does not seem efficient to use the decomposition algorithm as an evaluator routine for the movable activities problem.

Another straightforward extension is to a cost-time (CPM) divisible model. The usual CPM algorithm traces out the project cost versus duration by means of a parametric change in dual flow. This method is too laborious when divisible activities are added; however, since decomposition is being used, one could minimize total project cost, adding a constraint of fixed duration to the master problem (or vice-versa).

Other desirable extensions would be to include costs in a model with movable activities. For example, the movable tasks could be of fixed duration, but there would be cost-time tradeoffs available on ordinary activities; a CPM computation would be used for the evaluators on total project cost, and certain nodes of the arborescence would probably be truncated due to infeasibility with respect to project duration. In another direction, one could assign certain costs to placing a movable activity between a certain pair of nodes, or between a dummy pair; the resulting *optimal insertion model* could be used to solve cost-time critical path problems with piecewise-linear, but discontinuous and nonconvex costs. The essential difficulty here is to construct efficient evaluator functions.

REFERENCES

[1] Bertier, P. and B. Roy, "A Solution Procedure for a Class of Problems Having Combinatorial Character," ORC 67-34, Operations Research Center, University of California, Berkeley, (September 1967). A translation of "Une Procédure de Resolution Pour Une Classe de Problemes Pouvant Avoir Un Charactère Combinatoire," International Compution Center Bulletin, Vol. 4, pp. 19-28, (1965).

[2] Dantzig, G. B. and P. Wolfe, "Decomposition Principle for Linear Programs," Operations Research, Vol. 8, No. 1, pp. 101-111, (January-February 1960).

[3] Ford, L. K. and D. R. Fulkerson, FLOWS IN NETWORKS, Princeton University Press, Princeton, New Jersey, (1962).

[4] Jewell, W. S., "Divisible Activities in Critical Path Analysis," Operations Research, Vol. 13, No. 5, pp. 747-760, (September-October 1965).

[5] Jewell, W. S., "A Primal-Dual Multi-Commodity Flow Algorithm," ORC 66-24, Operations Research Center, University of California, Berkeley, (September 1966).

[6] Jewell, W. S., "Multi-Commodity Network Solutions," THÉORIE DES GRAPHES, ACTES DES JOURNÉES INTERNATIONALES D'ÉTUDES DE L'I.C.C., ROME, 1966, Dunod, Paris, (1967).

[7] Kelley, J. E., Jr., "Critical Path Planning and Scheduling," Operations Research, Vol. 9, pp. 296-320, (May-June 1961).

[8] Lawler, E. L. and D. E. Wood, "Branch-and-Bound Methods: A Survey," Operations Research, Vol. 14, No. 1, pp. 699-719, (July-August 1966).

[9] Matthyr, G., "Flot Optimum dans un Reseau à Capacités de Faisceaux," PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE ON OPERATIONAL RESEARCH, AIX-EN-PROVENCE, SEPTEMBER, 1960, English Universities Press, Ltd., London, pp. 164-170.

[10] Simonnard, M., LINEAR PROGRAMMING, Prentice-Hall, Englewood Cliffs, New York, (1966).

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D	
Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)	
1 ORIGINATING ACTIVITY (Corporate author) University of California, Berkeley	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP
3 REPORT TITLE DIVISIBLE AND MOVABLE ACTIVITIES IN CRITICAL PATH ANALYSIS	
4 DESCRIPTIVE NOTES (Type of report and inclusive dates) Research Report	
5 AUTHOR(S) (First name, middle initial, last name) William S. Jewell	

6 REPORT DATE November 1969	7a. TOTAL NO. OF PAGES 45	7b. NO. OF REFS 10
8a. CONTRACT OR GRANT NO DA-31-124-ARO-D-331 b. PROJECT NO 20014501B14C c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ORC 69-34 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
10 DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.		
11 SUPPLEMENTARY NOTES Also supported by the Office of Naval Research under Contract Nonr-222(83) and the Nat'l. Sci. Found. under Grants GP-8695 and GK-1684.	12 SPONSORING MILITARY ACTIVITY U.S. Army Research Office-Durham Box CM, Duke Station Durham, North Carolina 27706	
13 ABSTRACT SEE ABSTRACT.		

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811

Unclassified
Security Classification

Unclassified
Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Critical Path						
Linear Programming						
Decomposition Method						
Branch-and-Bound Method						

DD FORM 1 NOV 65 1473 (BACK)

Unclassified
Security Classification

A-31409